Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 793986, 12 pages
http://dx.doi.org/10.1155/2015/793986

Research Article

An Empirical Study of Greedy Kernel Fisher Discriminants

Tom Diethe

Department of Electrical and Electronic Engineering, University of Bristol, Bristol BS8 1UB, UK

Correspondence should be addressed to Tom Diethe; tom.diethe@bristol.ac.uk

Received 25 February 2015; Revised 6 May 2015; Accepted 14 May 2015

Academic Editor: Matteo Gaeta

A sparse version of Kernel Fisher Discriminant Analysis using an approach based on Matching Pursuit (MPKFDA) has been shown to be competitive with Kernel Fisher Discriminant Analysis and the Support Vector Machines on publicly available datasets, with additional experiments showing that MPKFDA on average outperforms these algorithms in extremely high dimensional settings. In (nearly) all cases, the resulting classifier was sparser than the Support Vector Machine. Natural questions that arise are what is the relative importance of the use of the Fisher criterion for selecting bases and the deflation step? Can we speed the algorithm up without degrading performance? Here we analyse the algorithm in more detail, providing alternatives to the optimisation criterion and the deflation procedure of the algorithm, and also propose a stagewise version. We demonstrate empirically that these alternatives can provide considerable improvements in the computational complexity, whilst maintaining the performance of the original algorithm (and in some cases improving it).

1. Introduction

Linear discriminant analysis and the related Fisher Discriminant Analysis were proposed by Fisher [1] as statistical approaches for classifying new data into two separate groups (the former assumes homoskedasticity, whereas the latter does not). The underlying assumption in Fisher Discriminant Analysis is that conditional probability density functions $p(\mathbf{x} \mid y = 1)$ and $p(\mathbf{x} \mid y = -1)$ are both normally distributed. Under this assumption, the Bayes optimal solution is to predict points as being from the second class if the ratio of the log-likelihoods is below some threshold (usually chosen as the point half-way between the class centroids).

Fisher Discriminant Analysis has been formulated using the "kernel trick," resulting in Kernel Fisher Discriminant Analysis (KFDA) [2, 3]. The resulting algorithm is Bayes optimal if conditional probability density functions of the data in the feature space $p(\phi(\mathbf{x}) \mid y = 1)$ and $p(\phi(\mathbf{x}) \mid y = -1)$ are normally distributed and has shown to be empirically competitive with other state-of-the-art algorithms such as the Support Vector Machine (SVM) [2, 4].

One drawback, as with most kernel methods, is that storing large kernel matrices is computationally prohibitive. In order to tackle this problem, one could subsample the dataset [5]. More interestingly, several authors have made attempts at addressing this issue by creating low rank kernel matrices behaving similarly to the full ranked ones whilst allowing for cheaper computations [6, 7]. Most important for us is the work of [8] where they devise a method of constructing low rank kernel matrices, motivated by a greedy approach called Matching Pursuit.

Matching Pursuit was proposed in the signal processing literature [9] as an attempt at finding a sparse set of basis functions (atoms) for a signal from a given dictionary and can be interpreted as a sparse version of least squares regression when the Orthogonal Matching Pursuit version is applied. In Orthogonal Matching Pursuit, each time a dictionary atom is chosen, the remaining weight vectors are projected into a space orthogonal to those chosen such that future atoms are only considered from a set far from those already picked. Kernel Matching Pursuit [10] has been proposed as the kernel counterpart of Orthogonal Matching Pursuit.

The greedy iterative idea of Matching Pursuit was applied to KFDA in order to impose "dual sparsity," as is achieved by the (kernel) SVM [4], resulting in the algorithm Matching Pursuit KFDA (MPKFDA) [11]. The authors showed that

this sparse version results in generalisation error bounds guaranteeing its future success. The bounds justify the choice of the greedy strategy, despite not being provably optimal [12], by ensuring that for any random choice of dataset and from any given distribution the resulting classifier will be "probably approximately correct" [13] with its predictions. In fact, the bound actually states that any strategy that simultaneously results in a sparse classifier and achieves a low training error will with high probability generalise well to new data, and given two classifiers with the same empirical error it favours the choice of the more parsimonious of the two.

One of the practical advantages of MPKFDA lies in the evaluation on test points, only k kernel evaluations are required (where k is the number of basis vectors chosen) compared to m (the number of samples) needed for KFDA. It is also worth stating that MPKFDA like KFDA has the advantage of delivering conditional probabilities of classification (unlike the SVM).

The paper has the following layout. Section 2 presents some recent developments in this topic. In Section 3.1 we present the notations used throughout the paper while Section 3.2 discusses the main practical contribution of the paper and presents the MPKFDA algorithm and its variants. The experiments are given in Section 3.4. The experimental results and discussion are in Section 4, and finally, some concluding remarks are given in Section 5.

2. Related Work

A greedy preimage algorithm similar in nature to MPKFDA was introduced by [14], and the comparisons in the paper MPKFDA was more accurate than the authors' method for 4 of the 6 datasets tested.

Following on from the empirical analysis given in [11], MPKFDA has since been applied to text classification [15], where experiments on the 20-Newsgroup dataset [16] demonstrated that MPKFDA maintained comparable classification accuracy compared with the SVM and k-nearest neighbours, whist significantly reducing the computation costs at prediction time.

Recently, an algorithm based on a manifold criterion and the Fisher criterion was, called Embedded Manifold-based Kernel Fisher Discriminant Analysis [17]. The authors claim that this preserves not only the local geometry structure of the data, but also the global discriminant structure of the data. This method bears striking similarities to the MPKFDA method of [11], except that whereas MPKFDA can be solved through an efficient iterative procedure, the method of [17] requires solving the full generalised eigenvalue problem.

From a theoretical perspective, a less general but tighter bound for KFDA than the type given in [11] has been developed [18], where the authors give a nontrivial, nonasymptotic upper bound on the classification error of KFDA under the assumption that the kernel induced space is a Gaussian Hilbert space. A more general compression bound on Matching Pursuit algorithms in a kernel defined feature space was developed by [19], which in principle could be extended to MPKFDA.

3. Materials and Methods

3.1. Preliminaries. Given a sample S containing m examples $\mathbf{x} \in \mathbb{R}^n$ and labels $y \in \{-1, 1\}$, let $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)'$ be the input vectors stored in matrix $\mathbf{X}$ as row vectors and let $\mathbf{y} = (y_1, \ldots, y_m)'$ be the labels in a column vector, where $'$ denotes the transpose of vectors or matrices. For simplicity, we assume that the examples are already projected into the kernel defined feature space, so that the kernel matrix $\mathbf{K}$ has entries $\mathbf{K}[i, j] = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$. The notation $\mathbf{K}[:, i]$ will denote the ith column of the matrix $\mathbf{K}$. When given a set of indices $\mathbf{i} = \{i_1, \ldots, i_k\}$ then $\mathbf{K}[\mathbf{i}, \mathbf{i}]$ denotes the square matrix defined solely by the index set $\mathbf{i}$. Given a Hilbert space $\mathscr{H}$, the reproducing property can be stated as $f(\mathbf{x}_i) = \langle f, \kappa(\mathbf{x}_i, \cdot) \rangle_{\mathscr{H}}$ for the reproducing kernel κ for every function $f(\mathbf{x}_i)$ belonging to $\mathscr{H}$.

3.2. Algorithmics. Firstly we review Fisher Discriminant Analysis and its kernel form. We then show how the (orthogonal) Matching Pursuit form of the algorithm is derived using the Nyström low-rank approximation method.

3.2.1. Fisher Discriminant Analysis. Using the notation from [3], the Fisher Discriminant Analysis problem can be written as

$$\mathbf{w}^* = \max_{\mathbf{w}} \frac{\mathbf{w}'\mathbf{X}'\mathbf{y}\mathbf{y}'\mathbf{X}\mathbf{w}}{\mathbf{w}'\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{w}}, \tag{1}$$

where $\mathbf{B} = \mathbf{D} - \mathbf{C}$ and $\mathbf{D}$ and $\mathbf{C}$ are given by

$$\mathbf{D}[i, i] = \begin{cases} \dfrac{2m^-}{m} & \text{if } y_i = +1 \\ \dfrac{2m^+}{m} & \text{if } y_i = -1, \end{cases}$$

$$\mathbf{C}[i, j] = \begin{cases} \dfrac{2m^-}{(mm^+)} & \text{if } y_i = +1 = y_j \\ \dfrac{2m^+}{(mm^-)} & \text{if } y_i = -1 = y_j \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where $m^+(m^-)$ are the number of positive (negative) examples.

3.2.2. Kernel Fisher Discriminant Analysis. In [3], it was shown that we can express $\mathbf{w}^*$ in the dual (unregularised) form as a linear combination of the training examples $\mathbf{w}^* = \mathbf{X}'\boldsymbol{\alpha}^*$, where $\boldsymbol{\alpha}^*$ is given by $\boldsymbol{\alpha}^* = ((1/\lambda)\mathbf{y} - \mathbf{B}\mathbf{X}\mathbf{w}^*)$, with λ being a Lagrange Multiplier. Assuming that the data has already been projected into a high dimensional feature space, the kernel matrix is defined simply as $\mathbf{K} = \mathbf{X}\mathbf{X}'$. This allows us to perform the so-called "kernel trick" and replace $\mathbf{w}$ with $\mathbf{X}'\boldsymbol{\alpha}$ to give the following dual form for Fisher Discriminant Analysis:

$$\boldsymbol{\alpha}^* = \max_{\boldsymbol{\alpha}} \frac{\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\mathbf{y}\mathbf{y}'\mathbf{X}\mathbf{X}'\boldsymbol{\alpha}}{\boldsymbol{\alpha}'\mathbf{X}\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{X}'\boldsymbol{\alpha}} = \max_{\boldsymbol{\alpha}} \frac{\boldsymbol{\alpha}'\mathbf{K}\mathbf{y}\mathbf{y}'\mathbf{K}\boldsymbol{\alpha}}{\boldsymbol{\alpha}'\mathbf{K}\mathbf{B}\mathbf{K}\boldsymbol{\alpha}}. \tag{3}$$

This kernel trick is based on the reproducing property, with the observation that in the equation to compute $\boldsymbol{\alpha}^*$ as well to

evaluate on a test point, all that is needed are the vectors $\mathbf{x}_i$ in inner products with each other. It is therefore sufficient to know these inner products only, instead of the actual vectors $\mathbf{x}_i$. This allows inner products between *nonlinear* mappings $\phi : \mathbf{x}_i \rightarrow \phi(\mathbf{x}_i) \in \mathcal{F}$ of $\mathbf{x}_i$ into a *feature space* $\mathcal{F}$, as long as the inner product $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)'\phi(\mathbf{x}_j)$ can be evaluated efficiently. In many cases, this inner product or *kernel function* can be evaluated much more efficiently than the feature vector itself, which can even be infinite dimensional in principle. A commonly used kernel function for which this is the case is the Radial Basis Function (RBF) kernel, which has a width parameter γ:

$$\kappa_\gamma\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-\frac{1}{\gamma}\left\|\mathbf{x}_i - \mathbf{x}_j\right\|^2\right). \tag{4}$$

3.2.3. Nyström Low-Rank Approximations. The Nyström method of low-rank approximation of the Gram matrix [20] is defined as

$$\widetilde{\mathbf{K}} = \mathbf{K}[:, \mathbf{i}]\, \mathbf{R}'\mathbf{R}\mathbf{K}[:, \mathbf{i}]', \tag{5}$$

where $\mathbf{R}$ is the Cholesky decomposition of $\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}$ such that $\mathbf{R}'\mathbf{R} = \mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}$. However, rather than using the full $[m \times m]$ low rank approximation, it would be preferable to work in the $[k \times k]$ space where $k \ll m$. In order to do this, we treat $\mathbf{K}[:, \mathbf{i}]\mathbf{R}'$ as a new input $\widetilde{\mathbf{X}}$ in Fisher Discriminant Analysis, which in effect means we are projecting into a k-dimensional subspace. Within this space, we can view

$$\widetilde{\mathbf{\Sigma}}_k = \widetilde{\mathbf{X}}'\widetilde{\mathbf{X}} = \mathbf{R}\mathbf{K}[:, \mathbf{i}]'\, \mathbf{K}[:, \mathbf{i}]\, \mathbf{R}', \tag{6}$$

as a form of covariance matrix within this space. This trick allows us to perform nonlinear discriminant analysis on a sparse subspace using standard (linear) Fisher Discriminant Analysis.

3.2.4. Matching Pursuit Kernel Fisher Discriminant Analysis. Orthogonal Matching Pursuit (nonorthogonal Matching Pursuit omits the deflation step) can be formalised as a general framework in machine learning, where we repeat the following steps:

(1) Function maximisation.

(2) Deflation.

We can have an Orthogonal Matching Pursuit algorithm for Fisher Discriminant Analysis [3] in the following way. Initially, we pick one example $\mathbf{i} = \{i_1\}$ and project the remaining training examples into the space defined by $\mathbf{i}$. We then find the index that maximises the KFDA criterion, after which we carry out a deflation of the kernel $\mathbf{K}$ to allow new training examples to be chosen. Finally, this give us a set $\mathbf{i}$ of training examples that can be used to compute the final weight vector $\mathbf{w}$, together with the Fisher Discriminant Analysis decision function $f(\mathbf{x}) = \operatorname{sgn}(\mathbf{w}'\mathbf{x} + b)$, where b is the bias and $\mathbf{x}$ an example.

We can define the following maximisation problem for a dual sparse version of Fisher Discriminant Analysis by setting $\mathbf{w} = \mathbf{X}'\mathbf{e}_i$, where $\mathbf{e}_i$ is the ith unit vector of length m, and substituting into the Fisher Discriminant Analysis problem described above (ignoring constants) to yield

$$\max_i \rho_i = \frac{\mathbf{e}_i'\mathbf{X}\mathbf{X}'\mathbf{y}\mathbf{y}\mathbf{X}\mathbf{X}'\mathbf{e}_i}{\mathbf{e}_i'\mathbf{X}\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{X}'\mathbf{e}_i} = \frac{\mathbf{K}[:, i]'\, \mathbf{y}\mathbf{y}'\mathbf{K}[:, i]}{\mathbf{K}[:, i]'\, \mathbf{B}\mathbf{K}[:, i]}. \tag{7}$$

Maximising the quantity above leads to maximisation of the Fisher Discriminant ratio corresponding to $\mathbf{e}_i$ and hence a sparse subset of the original KFDA problem. We would like to find the optimal set of indices $\mathbf{i}$. We proceed in a greedy manner (Matching Pursuit) in much the same way as [8, 10]. The procedure involves choosing basis vectors that maximise the Fisher Discriminant ratio iteratively until some prespecified number of k vectors are chosen.

After finding the best index i, the kernel matrix $\mathbf{K}$ is made orthogonal to the basis chosen by setting $\boldsymbol{\tau} = \mathbf{K}[:, i]/\|\mathbf{K}[:, i]\|$ and deflating using, for example, the projection deflation method [11, 21] $\mathbf{K} = (\mathbf{I} - \boldsymbol{\tau}\boldsymbol{\tau}')\mathbf{K}$. If $\boldsymbol{\tau}$ were "true" eigenvectors, this deflation ensures that remaining potential basis vectors will be chosen from a space that is orthogonal to those bases already picked. After choosing the k training examples, giving $\mathbf{i} = (i_1, \ldots, i_k)$, we can use the Nyström approximation defined in (6) to give us our new data matrix $\widetilde{\mathbf{X}} = \mathbf{R}\mathbf{K}[:, \mathbf{i}]'$. We then train Fisher Discriminant Analysis as in (1) in this new projected space to find a k-dimensional weight vector $\mathbf{w}_k$. Given a new point $\mathbf{z}$, using the kernel evaluated between the test point and the training points within the index set: $\mathbf{k} = \kappa(\mathbf{z}, \mathbf{x} \in \mathbf{i})$ and its projection into the Nyström subspace $\phi(\mathbf{z}) = \mathbf{R}\mathbf{k}'$, we can make predictions using the Fisher Discriminant Analysis prediction function:

$$f(\mathbf{z}) = \operatorname{sgn}\left(\left\langle \widetilde{\mathbf{w}}, \phi(\mathbf{z})\right\rangle + b\right). \tag{8}$$

3.2.5. Generalisation Error Bound for MPKFDA

Theorem 1 (generalisation error of MPKFDA [11]). *Let S be a sample of m points drawn independently according to a probability distribution P where R is the radius of the ball in the feature space containing the support of the distribution. Let $\widehat{\mu_k}(\mu_k)$ be the empirical (true) mean of a sample of $m-k$ points from the set $S \setminus \mathbf{i}$ projected into a k-dimensional space, $\widehat{\mathbf{\Sigma}}_k$ ($\mathbf{\Sigma}_k$) is its empirical (true) covariance matrix, $\mathbf{w}_k \neq 0$ with norm 1, and b_k is given, such that $\mathbf{w}_k'\mu_k \leq b_k$ and $\alpha \in [0, 1)$. Then, with probability $1 - \delta$ over the draw of the random sample, if $b_k - \mathbf{w}_k'\widehat{\mu_k} \geq \kappa(\alpha)\sqrt{\mathbf{w}_k'\widehat{\mathbf{\Sigma}}_k\mathbf{w}_k}$, then*

$$P\left(\mathbf{w}_k'\phi(\mathbf{x}) - b_k > 0\right) < 1 - \alpha, \tag{9}$$

where α solves the equation $\alpha = (b_k - \mathbf{w}_k'\widehat{\mu_k} - A)^2/(\mathbf{w}_k'\widehat{\mathbf{\Sigma}}_k\mathbf{w}_k + B + (b_k - \mathbf{w}_k'\widehat{\mu_k} - A)^2)$ such that

$$\begin{aligned}\left\|\widehat{\mu_k} - \mu_k\right\| &\leq \frac{R}{\sqrt{m-k}}\left(2 + \sqrt{k\ln\frac{em}{k} + 2\ln\frac{2m}{\delta}}\right),\\ \left\|\widehat{\mathbf{\Sigma}}_k - \mathbf{\Sigma}_k\right\|_F &\leq \frac{2R^2}{\sqrt{m-k}}\left(2 + \sqrt{k\ln\frac{em}{k} + 2\ln\frac{4m}{\delta}}\right).\end{aligned} \tag{10}$$

Proof. Here we present a sketch of the proof; for a full proof see [11].

First we need a bound the true and empirical means of an m sample S, which was given by [22], and the corollary of the bound on the true and empirical covariances, both of which make use of the radius of the ball in the feature space that contains the support of the distribution. The next ingredient is the lemma of [23] which bounds the error of a robust minimax classifier (whose maximiser coincides with the maximum of the KFDA objective) given the true mean and covariance. Combining these, and applying a k-dimensional projection (such as given by the Nyström method), gives us the result. □

3.3. Generalisations. We can see that the resulting bound does not require the optimisation criterion (7), nor does it require the deflation as proposed by the Nyström method. In fact, all it requires is that the classifier uses a sparse set of basis vectors, and that the training error is small. This allows us to consider other optimisation criteria and deflations methods, which potentially have similar (or even better) generalisation properties.

3.3.1. Alternative Optimisation Criteria. The optimisation criterion (7) scales as $\mathcal{O}(m^2)$, meaning that the algorithm scales as $\mathcal{O}(m^2k)$. It is worth investigating whether cheaper $\mathcal{O}(m)$ or even naïve $\mathcal{O}(1)$ methods can compete with this method. In addition, although at first sight the criterion might to make logical sense, there are other possible ways in which the sparse basis set could be chosen. Some alternatives are discussed below. The original optimisation criterion will be called `optimal` for the rest of the paper, with the short names for the other methods considered given in parentheses below.

(i) Pseudo-Fisher (`pseudo`): we know that the denominator in the Fisher criterion captures the "within-class" scatter of the data. In some cases, particularly in the case of balanced datasets, this may be assumed to be relatively uniform across the classes. Whilst of the same order, clear the maximisation will be significantly faster if only the numerator is used (essentially twice as fast). The maximisation is simply defined as

$$\max_i \rho_i = \mathbf{e}_i' \mathbf{X}\mathbf{X}'\mathbf{y}\mathbf{y}\mathbf{X}\mathbf{X}'\mathbf{e}_i = \mathbf{K}[:,i]'\,\mathbf{y}\mathbf{y}'\mathbf{K}[:,i]\,. \tag{11}$$

(ii) Random (`random`): of course one might ask if the Fisher criterion is really helping the optimisation at all. In the work of [11], it was simply assumed that the Fisher criterion was important in the optimisation. A simple test of the veracity of this is to simply select the bases uniformly at random (without replacement). This method is not quite as unprincipled as it may seem at first sight. By performing the optimisation in this manner, the algorithm reduces to becoming the randomised Nyström approximation [20], adapted for the KFDA framework.

There are two results for the Nyström approximation which are relevant: an upper bound on the expected reconstruction of the low rank matrix approximation [24] and a bound which shows that if there exists a separator with hard margin γ in the original space a Nyström projection of dimension $d \geq (8/\epsilon)[1/\gamma^2 + \ln(1/\delta)]$ will with probability $1-\delta$ over the selection of the d points defining the projection create a margin of at least $\gamma/2$ for all but at most an ϵ fraction of the training data [25]. The second statement implies the potential for good generalisation since a large margin classifier misclassifying some points has a provable bound on generalisation. Nonetheless it is not clear that this will be found by the margin maximising SVM, since it deals with margin errors using slack variables that do not simply count margin errors, let alone KFDA, which maximises the *average* margin. Furthermore, the assumption that there exists a hard margin separator in the original space is in practice unrealistic. A SVM solution with small objective might be found, implying good generalisation but at the expense of a number of points with nonzero slack variables. Nevertheless, the first statement, that the reconstruction error of the kernel matrix is bounded, implies that learning may be possible using this seemingly naïve method. Furthermore it provides a test that the Fisher criterion is worth computing in practice, and as such it is included in the experiments below.

(iii) Reverse Fisher (`reverse`): whereas the SVM finds points that are difficult to classify in each class and constructs a hyperplane that maximises the separation between these points, the MPKFDA seeks to find a few points that maximise average margin between the classes. One interesting, and again slightly perplexing possibility, is that the complete reverse of the Fisher criterion might be useful in the sparse selection of bases. The proposal here is not to completely reverse the KFDA algorithm but simply to use the reverse criterion for the selection of bases. By selecting bases in this way, the points that are least characteristic of the classes will be chosen first, which in high noise and/or highly nonlinear situations that require complicated decision borders could allow the algorithm to perform better for certain datasets. Of course, as the number of bases chosen approaches m, the algorithm reverts to the full Fisher discriminant. At the very worst, this provides another sanity check for the use of the standard criteria of (7). In this case, the optimisation criterion is

$$\min_i \rho_i = \frac{\mathbf{e}_i'\mathbf{X}\mathbf{X}'\mathbf{y}\mathbf{y}\mathbf{X}\mathbf{X}'\mathbf{e}_i}{\mathbf{e}_i'\mathbf{X}\mathbf{X}'\mathbf{B}\mathbf{X}\mathbf{X}'\mathbf{e}_i} = \frac{\mathbf{K}[:,i]'\,\mathbf{y}\mathbf{y}'\mathbf{K}[:,i]}{\mathbf{K}[:,i]'\,\mathbf{B}\mathbf{K}[:,i]}. \tag{12}$$

(iv) Reverse Pseudo-Fisher (`reverse-pseudo`): naturally, one could again consider a pseudooptimisation where only the between-class scatter is considered. The optimisation criterion then becomes

$$\min_i \rho_i = \mathbf{e}_i'\mathbf{X}\mathbf{X}'\mathbf{y}\mathbf{y}\mathbf{X}\mathbf{X}'\mathbf{e}_i = \mathbf{K}[:,i]'\,\mathbf{y}\mathbf{y}'\mathbf{K}[:,i]\,. \tag{13}$$

3.3.2. Deflation Methods. A matrix deflation modifies a matrix to eliminate the influence of a given eigenvector, typically by setting the associated eigenvalue to zero (see [21] for a more detailed discussion). For the greedy Fisher algorithm, the deflation step ensures that sufficiently different points are chosen at each step. It is easy to see that if a point gives a high value for the Fisher criterion, then nearby points will also give high values, and vice versa. By removing the influence of

a point, the scores for all nearby points should be lowered. This means that at the next step, points that are orthogonal (or close to orthogonal) will be selected. Of course this is easily tested by running the algorithm without any deflation step. This will be included in the experimental results for validation purposes (and will be called NONE).

Note that if we do not perform any deflation, the algorithm is simply performing subset selection for KFDA. In the case of the random optimisation criterion, we can see that this is equivalent to randomly selecting a subset of the data. The other optimisation criteria are then variations of greedy subset selection (see, e.g., [5]). We included experiments with no deflation, as well as the deflation methods discussed below.

In each of the methods, we assume that the basis vector with which the deflation will be performed has been normalised; that is, $\boldsymbol{\tau} = \boldsymbol{\tau}/\|\boldsymbol{\tau}\|$.

(i) Hotelling's Deflation (HOTELLING): in the Principal Component Analysis setting, the goal is to extract the leading k eigenvectors of the sample covariance matrix, $\boldsymbol{\Sigma} \succeq 0$, as its eigenvectors are equivalent to the loadings of the first k principal components. Hotelling's deflation method [21] is a simple and popular technique for sequentially extracting these eigenvectors. Here it is applied to the kernel matrix $\mathbf{K}$ rather than the covariance matrix $\boldsymbol{\Sigma}$. On the tth iteration of the deflation method, we would first extract the leading eigenvector of $\mathbf{K}$:

$$\boldsymbol{\tau} = \arg\max_{\boldsymbol{\tau}:\|\boldsymbol{\tau}\|=1} \boldsymbol{\tau}'\mathbf{K}\boldsymbol{\tau}, \tag{14}$$

and then use Hotelling's deflation to annihilate $\boldsymbol{\tau}$:

$$\mathbf{K} = \mathbf{K} - \boldsymbol{\tau}\boldsymbol{\tau}'\mathbf{K}\boldsymbol{\tau}\boldsymbol{\tau}'. \tag{15}$$

This procedure preserves annihilates a selected eigenvalue while maintaining all others, which also implies that it preserves positive-semidefiniteness. Sparse Principal Component Analysis [21] seeks to find sparse loadings which together capture the maximum amount of variance in the data, usually the additional constraint that the loadings are produced in a sequential fashion. Typically, Hotelling's deflation is applied by substituting an extracted "pseudoeigenvector" for a true eigenvector in the deflation step. Here we substitute the (normalised) vector found by the criteria (7). However, the properties of Hotelling's deflation, discussed in [21], depend crucially on the use of a true eigenvector, but we include the method for comparison.

(ii) Projection Deflation (PROJECTION): given the kernel matrix $\mathbf{K}$ and an arbitrary unit vector $\boldsymbol{\tau} \in \mathbb{R}^m$, an intuitive way to remove the contribution of $\boldsymbol{\tau}$ from $\mathbf{K}$ is to project $\mathbf{K}$ onto the orthocomplement of the space spanned by $\boldsymbol{\tau}$:

$$\mathbf{K} = \left(\mathbf{I} - \boldsymbol{\tau}\boldsymbol{\tau}'\right)\mathbf{K}. \tag{16}$$

If $\boldsymbol{\tau}$ is a true eigenvector, this reduces to Hotelling's deflation. In the general case, when $\boldsymbol{\tau}$ is not a true eigenvector, projection deflation maintains the desirable properties that were lost to Hotelling's deflation. For example, positive-semidefiniteness is preserved [21]. The deflation method was the one originally proposed for MPKFDA by [11].

(iii) Schur Complement Deflation (SCHUR): since the goal of the deflation step is to eliminate the influence a given basis vector, as measured through variance and covariances, it is reasonable to consider the conditional variance of the data variables given a pseudo-principal component. While this conditional variance is nontrivial to compute in general, it takes on a simple closed form when the variables are normally distributed. As KFDA assumes that the class conditional distributions are normal (in the feature space), this is not unreasonable here. The resulting deflation (as shown by [21]) is

$$\mathbf{K} = \mathbf{K} - \frac{\mathbf{K}\boldsymbol{\tau}\boldsymbol{\tau}'\mathbf{K}}{\boldsymbol{\tau}'\mathbf{K}\boldsymbol{\tau}}. \tag{17}$$

Schur complement deflation, like projection deflation, preserves positive-semidefiniteness, and also reduces to Hotelling's deflation if $\boldsymbol{\tau}$ is a true eigenvector.

(iv) Orthogonalised (Hotelling) Deflation (ORTHO-HOTELLING): while projection deflation and Schur complement deflation address the concerns raised by performing a single deflation using a non-eigenvector setting, difficulties arise when attempting to sequentially deflate a matrix with respect to a series of nonorthogonal pseudoeigenvectors.

A distinction must be made between the variance explained by a vector, and the additional variance explained given all previous vectors. These are equivalent in the Principal Component Analysis setting, as true eigenvectors are orthogonal, but in general, the vectors extracted by greedy methods such as MPKFDA will not be orthogonal. A modified version of Hotelling's deflation to account for this was given by [26]. Their procedure is equivalent to (15) for $t = 1$ and is expressed in terms of an iterative Gram-Schmidt decomposition for $t > 1$:

$$\begin{aligned} \mathbf{Q} &= \left[\mathbf{q}_1, \ldots, \mathbf{q}_{t-1}\right]', \\ \mathbf{q}_t &= \frac{\left(\mathbf{I} - \mathbf{Q}\mathbf{Q}'\right)\boldsymbol{\tau}}{\left\|\left(\mathbf{I} - \mathbf{Q}\mathbf{Q}'\right)\boldsymbol{\tau}\right\|}, \\ \mathbf{K} &= \mathbf{K} - \mathbf{q}\mathbf{q}'\mathbf{K}\mathbf{q}\mathbf{q}'. \end{aligned} \tag{18}$$

(v) Orthogonalised Schur Complement Deflation (ORTHO-SCHUR): finally we can consider a modification of the Schur complement deflation to account for the sequential noneigenvector deflation setting. As with the modified Hotelling method, at $t = 1$ the procedure is equivalent to the Schur complement method (17). For $t > 1$, we use $\mathbf{q}_t$ as defined in (18) and then apply the Schur complement deflation using $\mathbf{q}$:

$$\mathbf{K} = \mathbf{K} - \frac{\mathbf{K}\mathbf{q}\mathbf{q}'\mathbf{K}}{\mathbf{q}'\mathbf{K}\mathbf{q}}. \tag{19}$$

In Proposition 2.2 of [21] the authors show that, for the case of sparse Principal Component Analysis, applying this method will actually be equivalent to the standard Schur complement deflation. However, in the case of MPKFDA, the proof does not hold as the space spanned by all the previously extracted pseudoeigenvectors cannot be expressed as a linear combination of the previously chosen bases, as the pseudoeigenvalues are discarded.

Input: Kernel $\mathbf{K}$, training labels $\mathbf{y}$, sparsity parameter $T \geq 1$, number of bases to pick at each iteration $S \geq 1$.
(1) calculate matrix $\mathbf{B}$
(2) initialise $\mathbf{i} = \emptyset$
(3) **for** $t = 1$ to T/S **do**
(4) **for** $s = 1$ to S **do**
(5) $\{\mathbf{i}, i\} \leftarrow \arg\max_{i \notin \mathbf{i}} \rho_i$ (optimisation criterion)
(6) **end for**
(7) Deflate kernel matrix
(8) calculate the projection $\widetilde{\mathbf{X}} = \mathbf{R}\mathbf{K}[:, \mathbf{i}]'$ where $\mathbf{R}$ is the Cholesky decomposition of $\mathbf{K}[\mathbf{i}, \mathbf{i}]^{-1}$ and $\mathbf{i} = (\mathbf{i}_1, \ldots, \mathbf{i}_k)$
(9) **end for**
(10) train Fisher Discriminant Analysis using (1) in this new projected space to find a sparse weight vector $\widetilde{\mathbf{w}}$ and make predictions using (8)
Output: final set $\mathbf{i}$, (sparse) weight vector $\widetilde{\mathbf{w}}$, bias term b

Algorithm 1: Stagewise Greedy Kernel Fisher Discriminant Analysis.

3.3.3. Stagewise Optimisation. Stagewise Orthogonal Matching Pursuit [27] identifies all coordinates with amplitudes exceeding a specially-chosen threshold, solves a least-squares problem using the selected coordinates, and subtracts the least-squares fit, producing a new residual. After a fixed number of stages, it stops. In contrast to Orthogonal Matching Pursuit, many coefficients can enter the model at each stage in Stagewise Orthogonal Matching Pursuit while only one enters per stage in Orthogonal Matching Pursuit. The authors give numerical examples showing that Stagewise Orthogonal Matching Pursuit rapidly and reliably finds sparse solutions in compressed sensing, decoding of error-correcting codes, and overcomplete representation. We could employ the threshold selection strategy by selecting the set the new variables with $\rho_i > \tau$ for a given τ. However, the ρ_i does not directly correspond to the residuals found in regression, so for simplicity we employed a slightly different approach where we simply fixed the number of bases that could be included at each stage. The orthogonalisation is then performed with respect to all of the bases chosen at the last iteration. It remains as future work to prove theoretical guarantees for this method akin [27] or to find a more rigorous analog of the Stagewise Orthogonal Matching Pursuit application in this framework.

We give a more general version of MPKFDA in Algorithm 1 that includes the stagewise optimisation as well as allowing the choices of optimisation criteria and deflation methods defined above.

3.4. Experiments. We present a comparison on 13 benchmark datasets derived from the UCI, DELVE, and STATLOG benchmark repositories [28]. We analyse the performance of MPKFDA using RBF kernels as defined in (6), for each of the 5 optimisation criteria (`random`, `optimal`, `pseudo`, `reverse`, `reverse-pseudo`), for the 6 deflation methods (NONE, HOTELLING, PROJECTION, SCHUR, ORTHO-PROJECTION, ORTHO-SCHUR), and for 4 values of the stagewise optimisation method (1, 2, 5, 10), giving a total of 120 methods. The data comes in 100 predefined splits into training and test sets (20 in the case of the image and splice datasets) as described in [2]. For each of the datasets we used 5-fold cross-validation (c.v.) over the first five training splits to select the optimal RBF kernel width parameter γ using the original algorithm (i.e., the stagewise optimisation set to 1, the `optimal` criterion, and SCHUR deflation) with a range of values for γ: $2^{[-5,-3,-1,1,3,5]}$, selecting the median over the five sets as the optimal value. We then reran each algorithm using this value of γ to determine the best level of sparsity k (with the maximum value being set to $\min(200, m)$). We calculated the Fisher Discriminant Analysis validation error at intervals of 10 up to the maximum k in order to determine a good approximate sparsity level. It was deemed unnecessary to select a different γ for each algorithm, for computational reasons and comparability.

4. Results and Discussion

4.1. Stages = 1. It is quite difficult to compare the algorithms across all three dimensions (number of stages, optimisation criteria, and deflation methods) at once, so to begin with we focus on the standard setup where the stagewise method is not used (i.e., stages = 1). We performed a 3-way Analysis of Variance with the main effects of dataset, deflation type, and optimisation criterion. All main effects were significant ($p < 0.001$), so *post hoc* testing was done between margin means using Tukey's honestly significant difference test with $p < 0.001$. Figures 1, 2, 3, and 4 summarise the Average Error Rates (AERs), Average Standard Deviations (ASDs), and Average Running Times (ARTs) of the different methods. In Figure 1, the average error over all splits and all datasets is shown.

If we look first at the deflation methods, the last four methods (PROJECTION, SCHUR, ORTHO-PROJECTION, and ORTHO-SCHUR) have broadly similar performance in terms of error rate across the optimisation criteria, although the ORTHO-SCHUR method gives the best performance averaged over all of the criteria (AER = 0.218). Indeed, it is interesting to note that the best overall error (AER = 0.185) is achieved with the `pseudo` optimisation criterion when combined

FIGURE 1: Average Error Rate (AER): test errors averaged over all splits and all datasets, with stages = 1. Shorter bars indicate smaller error values, with the smallest overall error shown in bold text.

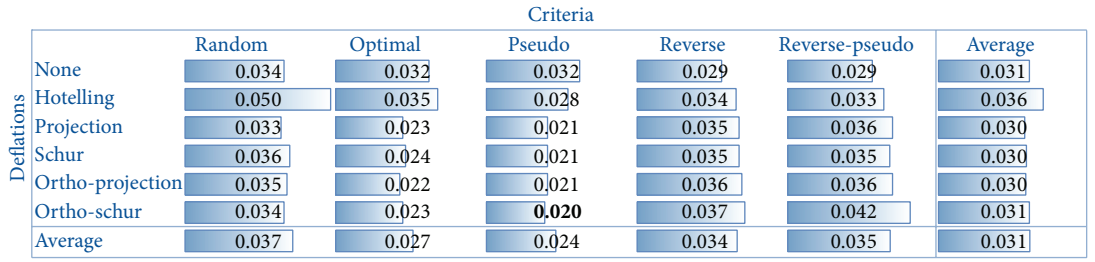

FIGURE 2: Average Standard Deviation (ASD): average over splits of the standard deviations of test errors averaged over datasets, with stages = 1. Shorter bars indicate smaller standard deviation values, with the smallest overall standard deviation shown in bold text.

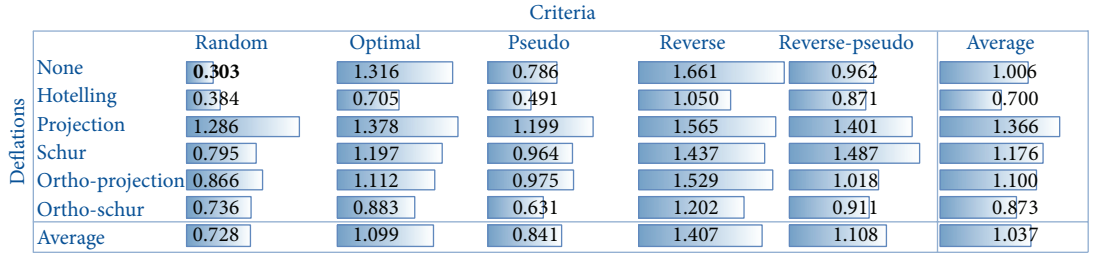

FIGURE 3: Average Running Time (ART): training time (seconds) averaged over all splits and all datasets, with stages = 1. Shorter bars indicate shorter computation time, with the shortest overall time shown in bold text.

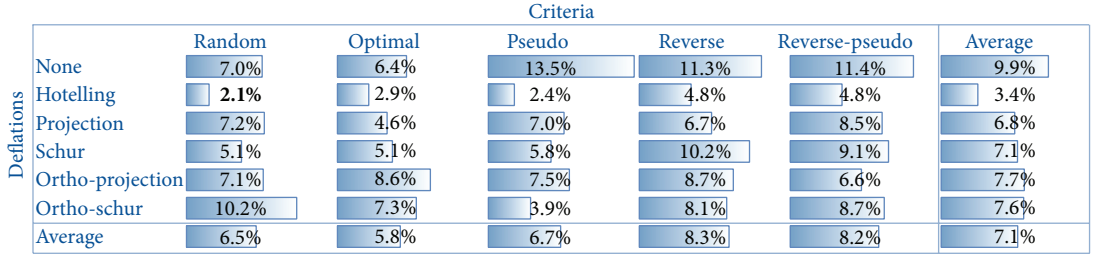

FIGURE 4: Average Sparsity (AS): sparsity level ($100(k/m)$) averaged over all splits and all datasets, with stages = 1. Shorter bars indicate increased sparsity, with the most sparse (overall) method in bold text.

with the ORTHO-SCHUR deflation method. In fact, when we examine the optimisation criteria, the `pseudo` method also gives the best performance when we average over all of the deflation methods (AER = 0.202), although it is not significantly different to the `optimal` method (AER = 0.215, $p > 0.05$). Note that averaging over the methods does not necessarily give the total indication of performance, as this may include some poor methods. Also note that the next best performing method (AER = 0.186) is actually the original method proposed by [11] (`optimal` combined with SCHUR).

The `reverse` and `reverse-pseudo` methods show the poorest overall performance and indeed are worse than

the `random` method. This would appear to confirm that the use of the Fisher Discriminant Analysis criterion and the pseudoversion (`optimal` and `pseudo`) are indeed sensible, which was supposed but never tested in [11].

For completeness, the average error for each dataset whilst varying the deflation methods with the optimisation method fixed to ORTHO-SCHUR and stages fixed to 1 is shown in Figure 7. The average error for each dataset whilst varying the optimisation methods with the deflation method fixed to `pseudo` and stages fixed to 1 is shown in Figure 8.

Of course the error rates cannot be taken in isolation. Figure 2 gives the Average Standard Deviations (ASDs), where the average is over the datasets and the standard deviation is over the test error of each of the splits. Here we note that the method that gave the best overall error (ORTHO-SCHUR with `pseudo`) also had the lowest overall standard deviation (ASD = 0.02), although not significantly different from several of the other methods ($p > 0.05$). This still indicates that the method is on average also the most stable across the splits of the data.

Next we examine the Average Running Times (ARTs) of the algorithms. The reported values in Figure 3 are the average running times (in seconds) taken for the final training. Note that this of course depends on the level of sparsity as well as the complexity of the method. Unsurprisingly, the fastest method is the one that uses no deflation and the `random` criterion, which corresponds to the random Nyström method applied to KFDA. The performance of this method is significantly worse (AER = 0.219 $p < 0.001$) than the best performing method and also has a significantly higher standard deviation (ASD = 0.034, $p < 0.001$) indicating that it is unstable. The `pseudo` criterion is the next fastest after random on average over the deflation methods, and in the case of the best performing deflation method for this criterion (ORTHO-SCHUR). The HOTELLING deflation method is the second fastest, but as can be seen in Figure 1 its performance is poor (e.g., for the `optimal` and `pseudo` criteria it is the worst method bar performing no deflation at all). The next fastest criterion is the ORTHO-SCHUR method (ART = 0.872 secs).

Finally, Figure 4 shows the Average Sparsity (AS) level (AS = $100(k/m)$) over all of the splits of all of the datasets for each of the methods. Of the deflation methods, the most sparse over all of the criteria is the HOTELLING method (AS = 3.4%), with the least sparse (again unsurprisingly) being no deflation (NONE, AS = 9.9%). Of the optimsiation criteria, the most sparse over all of the deflation methods is the `optimal` method (AS = 5.8%), followed by the `random` (AS = 6.5%) and `pseudo` (AS = 6.7%) methods. The most sparse individual method was HOTELLING and `random` (AS = 2.1%). Ignoring the HOTELLING method for its poor performing, the best performing method from Figure 1 (ORTHO-SCHUR and `pseudo`) is also the most sparse (AS = 3.9%).

4.2. Stages > 1. We now examine the performance of the stagewise version of the algorithm. Figure 5 shows a box plot of the average error over all datasets for the `pseudo` optimisation criterion with the ORTHO-SCHUR deflation method,

FIGURE 5: Stages (Average Error Rate): average error rates for varying the number of stages whilst keeping the criterion fixed (to `pseudo`) and the deflation method fixed (to ORTHO-SCHUR). The red line within each box indicates the median over the datasets.

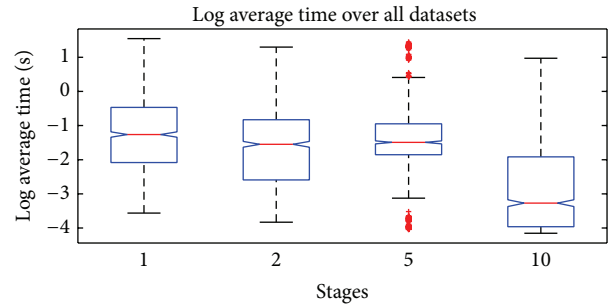

FIGURE 6: Stages (Average Running Time): log of average running time of varying the number of stages whilst keeping the criterion fixed (to `pseudo`) and the deflation method fixed (to ORTHO-SCHUR). The red line within each box indicates the median over the datasets.

whilst varying the number of stages [1, 2, 5, 10], (Figure 9 shows this for each dataset). Figure 6 shows a similar plot for the logarithm of the average running time. We observe a steady degradation of performance, until we reach 10 stages where the performance declines markedly. In fact the error rate over all the datasets is actually lower for stages = 2 (AER = 0.121, ASD = 0.009) than for stages = 1 (AER = 0.122, ASD = 0.009) whilst the computation time is clearly lower for stages = 2 (ART = 0.17 secs) than for stages = 1 (ART = 0.24 secs). The 95% confidence intervals for stages = 1 and stages = 2 overlap for the error rate, but not for the running time, indicating that stages = 2 is significantly faster whilst there is not a significant difference in performance. A similar pattern was observed with other deflation methods and optimisation criteria.

5. Conclusions

We provide extensive empirical analysis of a total of 120 variations on the MPKFDA algorithm of Diethe and Hussain [11]. The results indicate that whilst the method of [11] performs well, there are (statistically) significant improvements to be made in terms of computation time and generalisation performance by using different optimisation criteria for picking basis vectors and deflation methods. We found that

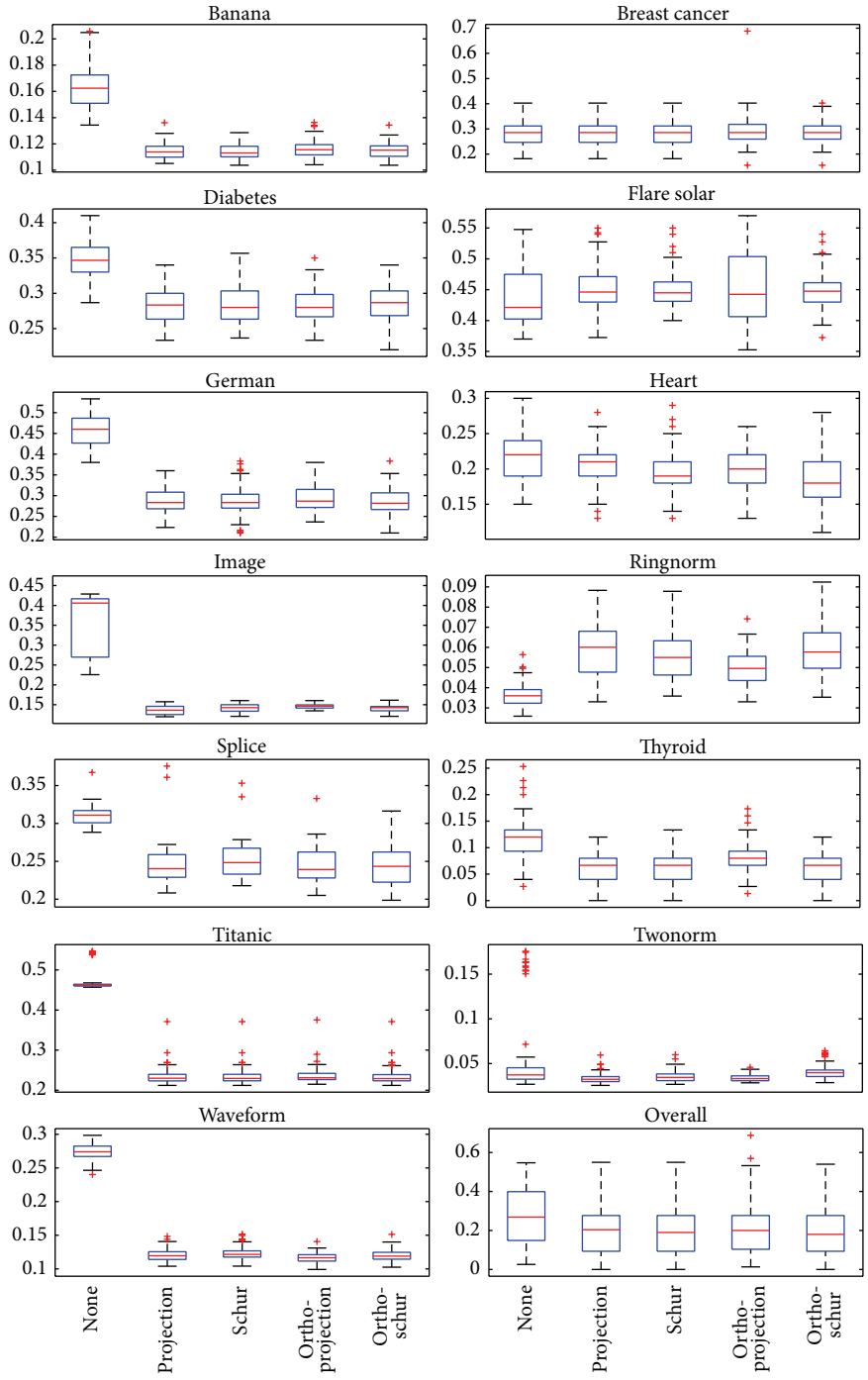

FIGURE 7: Deflations. Average error rates over the predefined splits for each dataset for varying the deflation methods with the number of stages fixed to one and the criterion fixed to `pseudo`. The red line within each box indicates the median over the splits.

the best performing method overall was a `pseudo` version of the Fisher Discriminant Analysis criterion (which only included the numerator) together with an orthogonalised version of the Schur complement deflation method. The fact that the pseudo criterion performed best was somewhat surprising and seems to indicate that the between-class scatter is not useful for selecting bases. We also analysed a stagewise version of the algorithm, where more than one basis vector could be selected during each iteration and showed that selecting two each time provided significant a speed-up whilst not affecting performance. It should be noted that these results are averaged over 13 datasets, and of course there

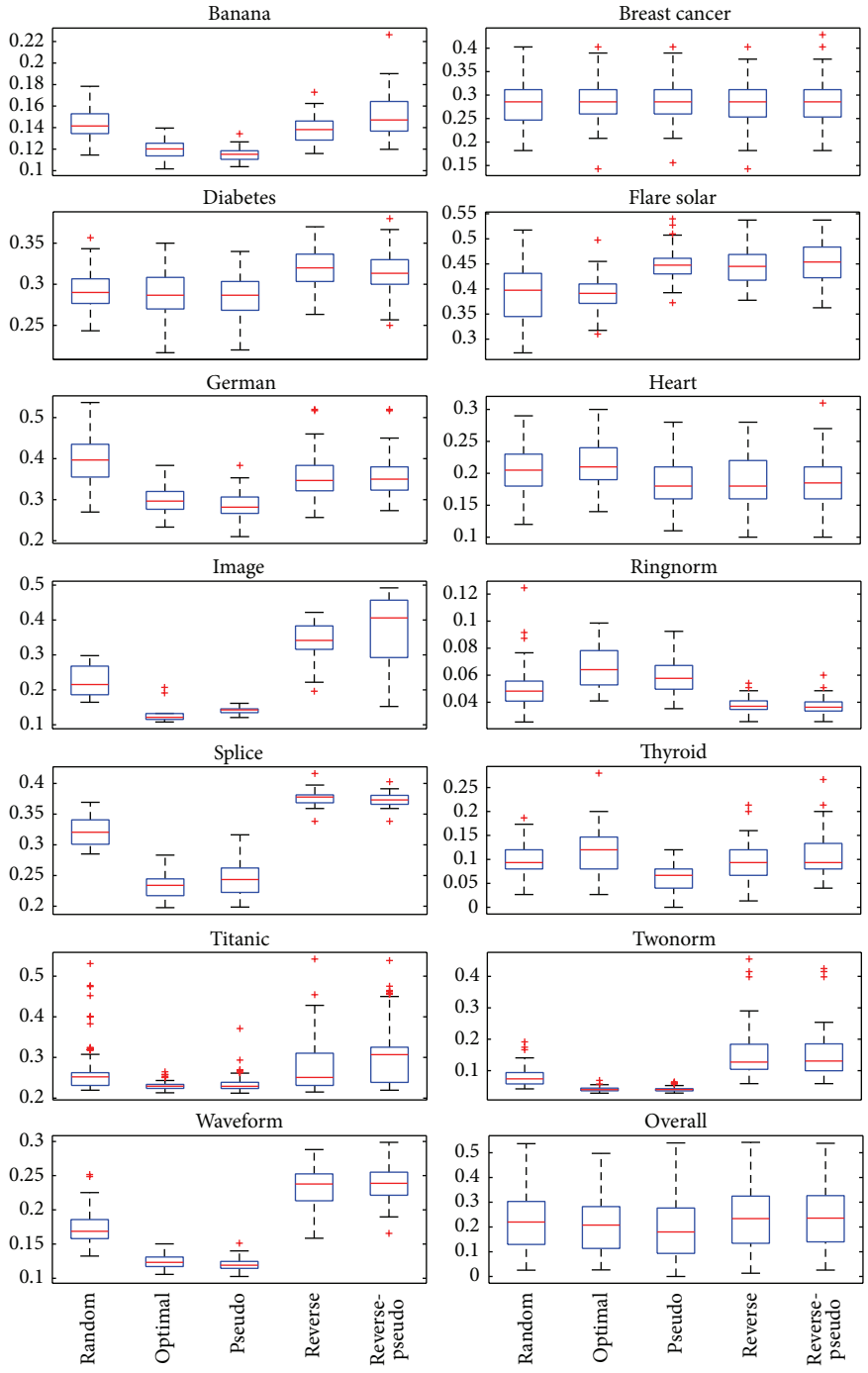

Figure 8: Criteria. Average error rates over the predefined splits for each dataset for varying the optimisation criteria with the number of stages fixed to one and the deflation method fixed to ORTHO-SCHUR. The red line within each box indicates the median over the splits.

maybe differences on individual datasets that are not clear from these results. However, the results give some guidance to the use of MPKFDA and its generalisations in practice.

Finally, it would be interesting to explore recent theoretical advances with respect to the KFDA algorithm [18] and Matching Pursuit algorithms operating in a kernel defined feature space [19], since a tighter bound for MPKFDA than the one given in [11] should be achievable.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

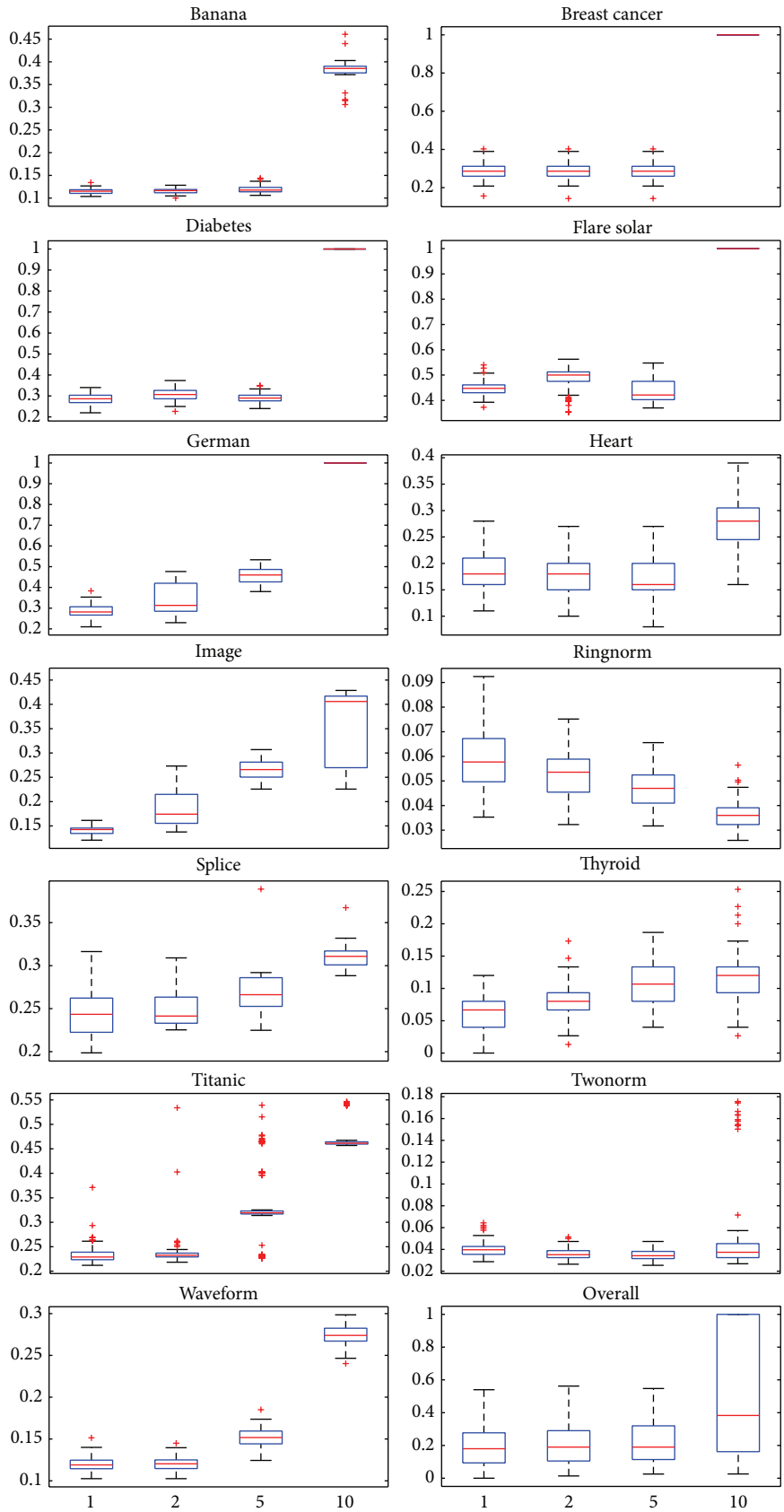

FIGURE 9: Stages. Average error rates over the predefined splits for each dataset for varying the the number of stages, the criterion fixed to `pseudo`, and the deflation method fixed to ORTHO-SCHUR. The red line within each box indicates the median over the splits.

Acknowledgment

This work was performed under the Sensor Platform for HEalthcare in a Residential Environment (SPHERE) Interdisciplinary Research Collaboration funded by the UK Engineering and Physical Sciences Research Council, Grant EP/K031910/1.

References

[1] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936.

[2] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. Müller, "Fisher discriminant analysis with kernels," in *Proceedings of the IEEE Neural Networks for Signal Processing (NNSP '99)*, E. W. Y. H. Hu, J. Larsen, and S. Douglas, Eds., pp. 41–48, IEEE, Madison, Wis, USA, August 1999.

[3] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, Cambridge, UK, 2004.

[4] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[5] S. A. Billings and K. L. Lee, "Nonlinear Fisher discriminant analysis using a minimum squared error cost function and the orthogonal least squares algorithm," *Neural Networks*, vol. 15, no. 2, pp. 263–270, 2002.

[6] F. R. Bach and M. I. Jordan, "Predictive low-rank decomposition for kernel methods," in *Proceedings of the 22nd International Conference on Machine Learning (ICML '05)*, pp. 33–40, ACM, New York, NY, USA, August 2005.

[7] B. Kulis, M. Sustik, and I. Dhillon, "Learning low-rank kernel matrices," in *Proceedings of the 23rd International Conference on Machine Learning (ICML '06)*, pp. 505–512, ACM, June 2006.

[8] A. J. Smola and B. Schölkopf, "Sparse greedy matrix approximation for machine learning," in *Proceedings of 17th International Conference on Machine Learning*, pp. 911–918, Morgan Kaufmann, San Francisco, Calif, USA, 2000.

[9] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[10] P. Vincent and Y. Bengio, "Kernel matching pursuit," *Machine Learning*, vol. 48, no. 1–3, pp. 165–187, 2002.

[11] T. Diethe and Z. Hussain, "Matching pursuit kernel Fisher discriminant analysis," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, vol. 5, pp. 121–128, April 2009.

[12] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[13] L. G. Valiant, "A theory of the learnable," *Communications of the Association of Computing Machinery*, vol. 27, no. 11, pp. 1134–1142, 1984.

[14] Q. Zhang and J. Li, "Constructing sparse KFDA using pre-image reconstruction," in *Neural Information Processing. Models and Applications*, pp. 658–667, Springer, Berlin, Germany, 2010.

[15] Q. Zhang, J. Li, and Z. Zhang, "Efficient semantic kernel-based text classification using matching pursuit KFDA," in *Neural Information Processing*, B.-L. Lu, L. Zhang, and J. Kwok, Eds., vol. 7063 of *Lecture Notes in Computer Science*, pp. 382–390, Springer, Berlin, Germany, 2011.

[16] K. Lang, "20 newsgroups data set," http://www.ai.mit.edu/people/jrennie/20Newsgroups/.

[17] G. Wang, N. Shi, Y. Shu, and D. Liu, "Embedded manifold-based kernel Fisher discriminant analysis for face recognition," *Neural Processing Letters*, 2014.

[18] R. J. Durrant and A. Kabán, "Error bounds for kernel Fisher linear discriminant in Gaussian Hilbert space," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS '12)*, pp. 337–345, La Palma, Spain, April 2012.

[19] Z. Hussain, J. Shawe-Taylor, D. R. Hardoon, and C. Dhanjal, "Design and generalization analysis of orthogonal matching pursuit algorithms," *IEEE Transactions on Transactions on Information Theory*, vol. 57, no. 8, pp. 5326–5341, 2011.

[20] C. K. I. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proceedings of the Advances in Neural Information Processing Systems*, vol. 13, pp. 682–688, MIT Press, 2001.

[21] L. Mackey, "Deflation methods for sparse PCA," in *Advances in Neural Information Processing Systems*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., vol. 21, pp. 1017–1024, MIT, 2009.

[22] J. Shawe-Taylor and N. Cristianini, "Estimating the moments of a random vector," in *Proceedings of the GRETSI 2003 Conference*, vol. 1, pp. 47–52, 2003.

[23] G. R. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M. I. Jordan, "A robust minimax approach to classification," *Journal of Machine Learning Research*, vol. 3, no. 3, pp. 555–582, 2003.

[24] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling techniques for the Nyström method," in *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics*, vol. 5 of *JMLR: W&CP 5*, AISTATS, 2009.

[25] A. Blum, "Random projection, margins, kernels, and feature-selection," in *Subspace, Latent Structure and Feature Selection*, vol. 3940 of *Lecture Notes in Computer Science*, pp. 52–68, Springer, Berlin, Germany, 2006.

[26] B. K. Sriperumbudur, D. A. Torres, and G. R. G. Lanckriet, "Sparse eigen methods by d.c. programming," in *Proceedings of the 24th International Conference on Machine Learning*, pp. 831–838, Morgan Kaufmann, San Francisco, Calif, USA, June 2007.

[27] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of un-derdetermined linear equations by stagewise orthogonal matching pursuit," Tech. Rep., 2006.

[28] T. Diethe, 13 Benchmark datasets derived from the UCI, DELVE and STATLOG repositories, 2015, https://github.com/tdiethe/gunnar_raetsch_benchmark_datasets/.